Kinect@Home: Crowdsourced RGB-D data

Rasmus Göransson
Centre for Autonomous Systems
KTH Royal Institute of Technology
SE-100 44 Stockholm, SWEDEN
Email: rasmusgo@kth.se

Alper Aydemir
Computer Vision Group,
Jet Propulsion Laboratory, NASA
Los Angeles, CA 91109 USA
Email: Alper.O.Aydemir@jpl.nasa.gov

Patric Jensfelt
Centre for Autonomous Systems
KTH Royal Institute of Technology
SE-100 44 Stockholm, SWEDEN
Email: patric@kth.se

***Abstract*—Algorithms for 3D localisation and mapping and 3D reconstruction are getting increasingly mature. In this paper we argue that it is time to also make the dataset on which they are tested more realistic to reflect the real world in the homes of real people rather then testing on data gathered in the lab or at best in a few places, and almost always by the people that designed the algorithm. We present the first RGB-D dataset from the crowd sourced data collection project Kinect@Home which contains 54 challenging recordings. We also present an evaluation of two different algorithms, the Kinfu algorithm and a key point based method. Our experimental evaluation highlights the different characteristics and error modes of the two methods.**

I. Introduction

Publicly available datasets has enabled rapid progress in applications such as simultaneous localization and mapping (SLAM) and visual recognition. Especially in computer vision, benchmarking challenges based on image datasets helps to determine the state-of-the-art methods and new datasets are put forward and adopted [1], [2]. These datasets help generating reproducible research in the field and allow benchmarking for objective comparisons of new methods against the state-of-the-art. In robotics, dataset such as the Victoria Park [3] and the Radish datasets [4] have allowed the community to depart from each paper using its own dataset. The RAWSEEDS project [5] focused on this issue. It was funded by the European Union which shows that collecting data is acknowledged as important.

Since their introduction, RGB-D cameras such as the Microsoft Kinect has seen widespread adoption in robotics research due to their ability to generate reliable 3D data at a fast frame rate at low cost. As a result of this, most robots are equipped with RGB-D cameras today.

Robotics today is close to deliver useful robots that can perform a variety of tasks in everyday environments, which necessitates skills such as 3D mapping and localization, navigation and object manipulation. What would our robots be faced with if they were to be shipped now? Most likely the robots would be taken by surprise at the complexity and variability of the environment. Most, if not all, datasets that our algorithms are tested on are collected in one or at best a few sites and almost always by the same people that have developed the algorithms and that will perform the evaluation. This creates several forms of biases. The designer of the algorithm knows what the weaknesses and strengths are and will, weather he wants to or not, and in one way or the other, adapt the way the data is collected. The environments that are used for tests are environment selected by the same designer and not the environment where said robots would be deployed. So how do we get data from the wild, from the real world collected by non-researchers and lots of it?

As mentioned, the Kinect is a commonly used robotics sensor but more importantly it is in the living rooms of millions of houses. This poses an opportunity to improve robotics research if we can put them to good use. In this paper, we tackle the challenge of building objective benchmark datasets which reflects the complexity of the real world and we do that using crowd sourcing. Crowd sourcing has been used for a number of different tasks. One of the main players in this area is Amazon that launched its Mechanical Turk in 2005. Here so called *Human Intelligence Tasks* (HIT) can be defined and people can earn money by solving these tasks which could be to label images as in ImageNet [1] or videos as in VATIC [6]. Our endeavour, Kinect@Home[1] [7], is similar to that of the Berkeley 3-D Object Dataset [8], but whereas they only collect single RGB-D images which severely limits the use in robotics we collect RGB-D video sequences.

The main contribution of this paper is the presentation and analysis of the first dataset from the Kinect@Home project. We also provide an evaluation of two different types of pose estimation methods on this data. The outline of the paper is as follows. In Section II we briefly discuss the process of gathering the data. In Section III we describe two different 3D SLAM methods used in the analysis of the data; the kinfu algorithm and a key point based method. In Section IV we present an experimental evaluation and a thorough analysis of the dataset itself. In Section V we summarise our work. For details beyond the limited space of this paper please refer to [9].

II. Data collection method

Two major challenges when collecting data from the general public for helping robotics research is to make the process simple enough and to motivate people to take the time to actually do it. To address the first issue we designed a browser based system. A browser plugin compresses the data and sends it to our servers continuously while the user records the data. If the user has a fast internet connection most of the data will have been uploaded when the user stops gathering data. The process is started and stopped by single clicks in the browser. For more technical details see [7].

To motivate the user to upload data we build a 3D model of the data and give it back in return. These models can be

[1] http://www.kinectathome.com

shared with others, embedded in webpages, etc. Admittedly, this type of motivation appeals mostly to early adopters of new technology but given the feedback and interest in media it seems to have served the purpose to raise interest and generate data. The 3D reconstruction is currently performed without any user interaction. Allowing for user interaction could result in better models but would guide the users towards collecting data that the current reconstruction software can handle resulting in less data that exposes the weaknesses.

III. 3D MAPPING

To analyse the Kinect@Home dataset we have looked at the task of 3D reconstruction, by which we mean the task of not only aligning the point clouds from the individual RGB-D frames but also estimating the geometry and assigning a texture to it. This corresponds to the task that needs to be solved to be able to return 3D models to the users uploading data. Ideally, each dataset would be accompanied with the ground truth like in [10] but this is not possible when the data is collected by the general public. Therefore, as a means to assess the difficulty of the recorded data and the performance of the algorithm we have chosen to look at the end result of the 3D reconstruction.

Figure 1 illustrates the pipe line for the reconstruction. The first step is to estimate the camera pose of the individual RGB-D frames (i.e. SLAM). The second step creates a geometric model which in the last step can be coloured / textured. Figure 1 also indicates what type of data that is used for the different steps. In general both depth and RGB data is used for the camera pose estimation, whereas only depth data, i.e. the geometric information is used in the geometric reconstruction given the estimated camera poses. Texturing makes use of both RGB and depth information.

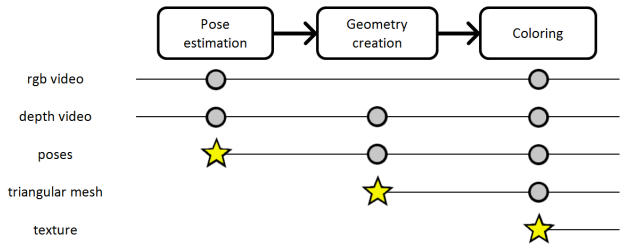


Fig. 1. The reconstruction process is divided into three parts. Each part processes data from previous steps. The stars indicate writing data and the circles indicate reading data.

A. Pose estimation

We compare two separate versions of pose estimation. An error in the pose estimation will result in bad geometry and reconstruction. As the two method share the rest of the pipeline, differences in the end result can be traced to the start. The first pose estimation method is based on SURF features and the second is a modified version of the open source implementation of Kinect Fusion [11] available as PCL Kinfu [12].

- **SURF based pose estimator** The poses are estimated by building a sparse map of SURF features. An initial sequential frame-to-frame pass is followed by g2o ([13]) to reduce drift. RANSAC is used to find matches from candidates with similar features also taking into account geometric consistency. Keypoints without depth information as well as depth information without keypoints are discarded. This makes the method weak to areas with few keypoints or missing depth data.
  Partial loop closing is done by matching frames from different points in time, not only consecutive frames. This makes the estimation more robust and precise.

- **Kinfu based pose estimator** PCL Kinfu comes as an interactive application which fetches the frames directly from the Kinect without intermediate storage. This implementation has been modified to instead load the frames from disk and store all the estimated poses. The interactive parts have been removed to allow the program to be run by a script without user interaction. The reconstructed volume is by default limited to 3x3x3 meters which is too small for many recordings. This causes problems when the user moves the camera too far as there will be no map and the tracking gets lost. Kintinuous [14] by Wheelan et.al solves this by moving the volume and the information within it to keep up. In this work we have solved this problem in a more crude way simply by resetting the volume completely when the camera moves too far. All the geometry created while tracking is discarded by these resets. This is possible since we are not after the geometry at this stage but only the poses.

TABLE I. SUMMARY OF CHARACTERISTICS OF THE TWO POSE ESTIMATION METHODS.

	SURF based		Kinfu based
+	Utilize rgb data	-	Discard rgb data
-	Discard most depth data	+	Utilize most depth data
-	Rely on structure in rgb	+	Indifferent to structure in rgb
+	Indifferent to structure in depth	-	Rely on structure in depth
+	Bundle adjustment	-	Only looks backwards in time

Note that the Kinfu based pose estimator does not use any RGB data, only depth data. The SURF based estimator on the other hand discards most of the depth data since it only reads the depth of the keypoints. This leads to different weaknesses and strengths of the two methods:

The SURF based estimator depends heavily on the availability of keypoints whereas the Kinfu based method depends on having structure. If we point the camera at a white flat wall the Kinfu based method would be able to estimate the distance to the wall and the orientation of the normal very well. However, the translation along the plane and the rotation around the normal cannot be estimated at all, i.e. only three out of six degrees of freedom can be estimated. The keypoint based method would not be able to estimate the pose at all. If we have three keypoints or more the SURF based method is able to estimate the full pose. Table I summarises the differences. This illustrates the core difference between the two methods, the result of which we will see in the experimental evaluation.

B. Reconstruction of geometry

The core component of the geometric reconstruction step in our pipe line is the Truncated Signed Distance Function

Fig. 2. Four examples from the Kinect@Home dataset. Notice the large variation in type and size of scene, structure, texture, etc. All natural variations of the real world that our robots need to be able to deal with.

(TSDF). This is a volumetric representation where the geometry is stored in a grid where each cell gives the (signed) distance to the closed surface. The distance function is positive in free space and negative in occupied space. The surfaces are found by solving for the zero crossings of the distance functions. This eliminates the discretisation effects of the grids assuming that the structure that we try to model is large enough. One of the big advantages of the TSDF is that it allows us to combine all depth data from all registered scans, in a straightforward manner. The TSDF is independent on how the poses for the scans where found and hence does not depend on having, for example, keypoints to reconstruct the geometry. The TSDF volume is created in the GPU and requires a lot of memory as it is a dense volume. This limits the size of the space that we can deal with. To overcome this limitation, the total volume to be reconstructed is divided into a uniform grid of 3x3x3 meter parts. Each part is reconstructed using the poses previously calculated. The depth data is loaded from disk into GPU memory and smoothed to reduce the influence of noise before being integrated into the TSDF volume. Only the depth images corresponding to poses where the bounds of the camera frustum intersect the current TSDF volume are loaded. The marching cubes algorithm [15] is used to extract a triangular mesh from each TSDF volume one at a time. These triangular meshes are downloaded from the GPU and stored in system RAM. As the parts are aligned in a uniform grid, stitching the meshes becomes easier because they are already aligned without overlap. The merged mesh is decimated based on the principle of edge collapses.

C. Texturing

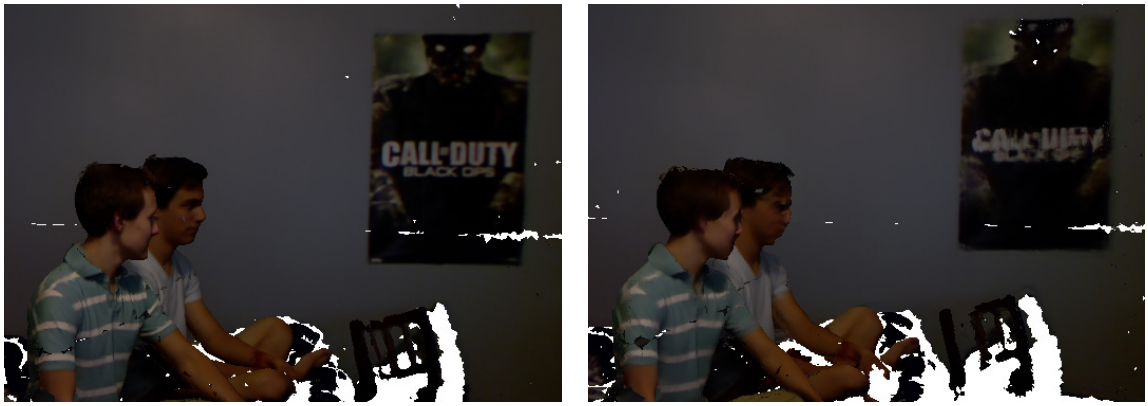
The texture information from each frame is combined in texture space, pixel by pixel, in an iterative fashion. The average colour for each pixel is computed. The distance from each sample to the average is then used to adjust the weight of the sample. Samples that are further away from the mean are given lower weight. A new weighted average is computed and the process is repeated 20 times. This results in an approximation of the median in three dimensions. This effectively reduces the influence of samples that are outliers, such as samples from the border of foreground objects on the background, occasional shadows, and specularities. The unpainted areas of the texture is filled with the colour from nearby pixels in a breadth first order. This removes artifacts near seams and fills in missing data.

IV. Experimental evaluation

Since most of the recordings are made by users with little or no knowledge of the reconstruction system, the data is unpredictable and contain lots of challenges. At the time of evaluation, the Kinect@Home dataset contained around one thousand recordings.

A. Dataset

A subset of 54 recordings were selected for analysis. The recordings were selected to give reasonable restrictions and variation in the data. The system is not designed to reconstruct dynamic scenes, so recordings with lots of moving objects were discarded. Small movements, such as shadows, rotating fans, eyes moving and blinking were accepted. Recordings that were very short or did not contain any movement of the camera were not considered since they lack interesting information. The subset is also chosen to contain a large variety in scenes and objects, both in size and type. Figure 2 shows four example datasets which have been reconstructed. Notice the large variation in size and type of scene.



(a) SURF (b) Kinfu

Fig. 3. The textures of models using poses from the SURF based pose estimator were often sharper and with better geometry as in (a) compared to (b).

Fig. 4. This model was built with the SURF based pose estimator. The wall indicated by the arrow has duplicated surfaces because of errors in the pose estimation. This problem did not appear with the Kinfu based estimator.

TABLE II. SUMMARY OF POSE ESTIMATION RESULTS FOR THE 54 SELECTED RECORDINGS.

Pose estimator	Good	OK	Poor
SURF based	26	11	17
Kinfu based	31	3	20

B. Evaluation of results

Figure 2 shows four successful reconstruction examples. However, there are many things that can go wrong during reconstruction. The 54 recordings were processed twice. Once with the SURF based pose estimator and once with the Kinfu based estimator. The resulting models were inspected and graded into one of three categories:

- **Good** - No easily noticeable errors.
- **OK** - Only minor errors typically visible as duplicate surfaces on walls.
- **Poor** - Severe errors leading to duplicated or missing objects, or even a total mess of polygons.

A summary of the results is shown in table II. The full listing with results for each individual recording is available in table III. It should be noted that the poses estimated using SURF gave noticeably sharper textures even when both sets of poses were considered “good”. See figure 3 for an example. On the other hand, duplicated surfaces on walls that do not appear with the Kinfu poses are common with the SURF poses. See figure 4 or an example.

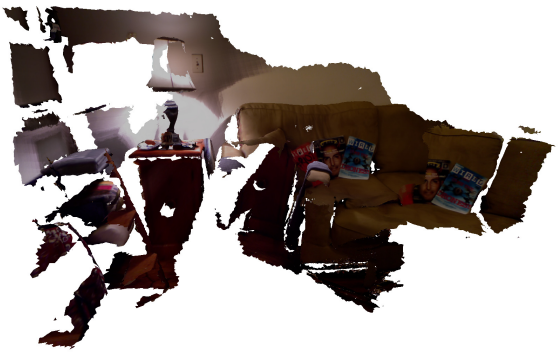


Fig. 5. A problematic reconstruction from a living room.

Four of the 54 recordings are presented and analyzed in more detail in the following subsections to show what happens when things go wrong and what happens when things go right.

1) Living room: A reconstruction with several interesting problems can be seen in figure 5. Three common problems are clearly visible:

- **Pose estimation error** - This can be seen from the ”extra” sofa to the right (there should only be one stack of magazines). The pose estimation gets lost when the user is filming too close to the sofa near the floor. The camera is so close that there are not enough features with depth data for the keypoint-based pose estimation to keep track. When the camera moves back again, the sofa gets registered a second time but now in a different position. If the loop closing algorithm was better, it would realize that the two sofas are actually the same and adjust the camera poses accordingly. As SLAM is an active area of research these problems are addressed by many people already. What we can also see looking at all data is that a common problem is that people move the camera too close to objects.
- **Holes in the model from missing data** - Since the Kinect sensor uses an infra-red pattern to estimate depth, it has to see the pattern to produce depth

Fig. 6. A Christmas tree is an example of a difficult object for reconstruction.

data. The hole behind the lamp is there because of occlusion. The hole in the screen of the lamp comes from the infra-red pattern being saturated by the infra-red light from the lamp itself. These holes can likely be filled in, at least partially, by using motion stereo from the RGB images.

- **Change in automatic rgb camera settings** - When the user moves the camera towards the lamp to the left, the camera darkens the whole image. This results in a sudden change of brightness on the wall. If the exposure settings would be estimated and adjusted for throughout the video, this problem could be reduced or canceled completely.

2) Christmas tree: Another interesting problematic scene is shown in figure 6. This scene contains a Christmas tree. The system presented in this paper is designed with certain types of scenes in mind. Detail in color is handled much better than detail in geometry. The branches with needles are very unsuitable for each step of our process. The Kinect sensor is unable to register thin objects like needles individually. The TSDF volume would not be able to represent them as individual objects even if they were recorded properly. The polygonal mesh would need extremely many polygons to represent every needle. The coloring step would have a hard time unwrapping all the surfaces and painting the texture.

What really happens with this recording is that the Kinect sensor lumps the needles together. The areas between the lumps are missing depth data because of all the occlusions. The lumps are actually decently represented by the TDSF volume but the true geometry has already been lost. The decimation cannot simplify the model very much since it is so fragmented by the branches. When comparing the model to the original tree in the video sequence, the video sequence shows lots of small Christmas tree lights which are missing in the reconstruction. This can be explained by missing depth data due to weaknesses of the Kinect depth sensor as previously mentioned.

3) Kitchen corner: Figure 7 shows a kitchen corner with many polygons even though the geometry seems to be smooth in reality. This is a problem that occurs in many recordings because noisy geometry fails to get simplified by the decimation algorithm. Pose estimation errors often generates bad geometry but the problem occurs even when the poses are near perfect.

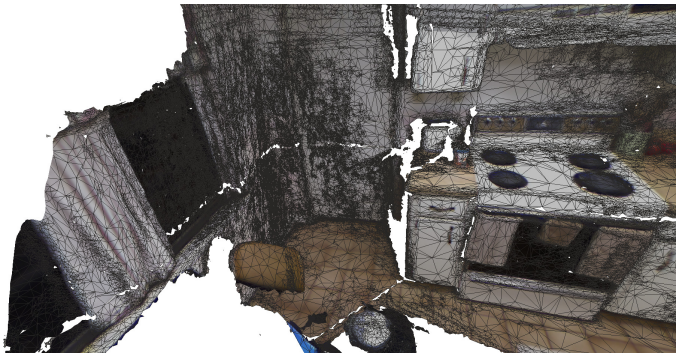


Fig. 7. This model is rendered with a wireframe to show the polygon structure. Notice that the corner has excessively many polygons. This is an effect of noisy depth data. The noise generates uneven surfaces which does not get simplified by the decimation algorithm.

Fig. 8. Reconstruction of a desktop environment.

Geometry only seen at a distance becomes noisy because the noise in the depth data from the Kinect increases with distance. The reconstruction system fights the noise by smoothing the depth data. The amount of smoothing is a tradeoff between preserving detail and removing noise. Handling this noise better could have a great impact on polygon counts for many models.

4) Desktop environment: A good result can be seen in figure 8. The model is almost complete with only little data missing in some places. The textures are very sharp with individual keys clearly visible on the keyboards and sharp lines in the drawings on the wall. The pose estimation has succeeded to generate very precise poses which makes the following steps produce good geometry and texture.

V. Summary and Conclusions

In this paper we have presented a crowdsourced RGB-D dataset for evaluation of algorithms that operate on RGB-D data. The dataset consists of 54 individual recordings gathered in the Kinect@Home project. The recordings have been made by the general public. We also present an analysis of the dataset based on the results of 3D reconstruction experiments using two different ways to estimate the motion of the RGB-D sensor, one based on keypoints and the other based on the Kinfu algorithm. The analysis strengthens our claim that real world data is challenging and that robotic systems need to be tested on this type of data to work in the wild.

A. Future work

We are currently in the process of preparing the dataset for public use. It will become available at http://www.kinectathome.com/datasets. The experiments also suggest that combining the two methods would increase the performance. Mechanisms to encourage users to acquire data of certain kinds, such as certain scenes or containing certain objects to make the dataset richer would be interesting to investigate.

ACKNOWLEDGMENT

This work was funded by SSF through its Centre for Autonomous Systems, VR under contract C0475401, the School of CSC at KTH Royal Institute of Technology, the EU FP7 project CogX and the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 (STRANDS).

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[3] J. Guivant, E. Nebot, and S. Baiker, "Autonomous navigation and map building using laser range sensors in outdoor applications," *Journal of robotic systems*, vol. 17, no. 10, pp. 565–583, 2000.

[4] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[5] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Auton. Robots*, vol. 27, no. 4, pp. 353–371, 2009.

[6] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, pp. 1–21, 10.1007/s11263-012-0564-1. [Online]. Available: http://dx.doi.org/10.1007/s11263-012-0564-1

[7] A. Aydemir, D. Henell, P. Jensfelt, and R. Shilkrot, "Kinect@home: crowdsourcing a large 3d dataset of real environments," in *2012 AAAI spring symposium series*, 2012.

[8] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3d object dataset: Putting the kinect to work," in *Consumer Depth Cameras for Computer Vision (CDC4CV) workshop*. Springer, 2011, pp. 141–165.

[9] R. Göransson, "Automatic 3d reconstruction with kinect : A modular system for creating high quality light weight textured meshes from rgbd video," Master's thesis, KTH Royal Institute of Technology, School of Computer Science and Communication, 2013.

[10] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[11] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, 2011, pp. 127–136.

[12] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.

[13] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[14] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially extended kinectfusion," Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, Massachusetts, Tech. Rep. MIT-CSAIL-TR-2012-020, July 2012. [Online]. Available: http://hdl.handle.net/1721.1/71756

[15] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.

TABLE III. POSE ESTIMATION RESULTS FOR THE 54 SELECTED RECORDINGS.

Number	SURF	Kinfu
1	good	good
2	good	good
3	good	good
4	good	good
5	good	good
6	good	good
7	good	good
8	good	good
9	good	good
10	good	good
11	good	good
12	good	good
13	good	good
14	good	good
15	good	good
16	good	good
17	good	good
18	good	good
19	good	good
20	good	good
21	good	good
22	good	ok (minor error to the left)
23	good	ok (minor error)
24	good	poor (background misaligned)
25	good	poor (corner and monitor)
26	good	poor (lost tracking)
27	ok (minor error)	good
28	ok (minor error on wall)	good
29	ok (minor error on wall)	good
30	ok (minor errors)	good
31	ok (minor errors)	poor
32	ok (minor error)	poor (lost track)
33	ok (minor errors)	poor (lost track)
34	ok (minor error on wall)	poor (lost track)
35	ok (minor error)	poor (lost tracking)
36	ok	poor (lost tracking)
37	ok (shaky)	poor (stopped tracking)
38	poor (lost tracking)	good
39	poor (floor misaligned)	good
40	poor (floor misaligned)	good
41	poor	good
42	poor (slanting wall)	good
43	poor (lost tracking)	good
44	poor (lost tracking)	ok (duplicate wall)
45	poor	poor
46	poor	poor
47	poor	poor
48	poor	poor
49	poor	poor
50	poor (complete mess)	poor (complete mess)
51	poor (lost tracking)	poor (lost tracking)
52	poor (lost tracking)	poor (lost tracking)
53	poor (lost tracking)	poor (lost tracking)
54	poor (lost tracking)	poor (lost tracking)